$n^2$ Sorts

• Lots of $\mathcal{O}(n^2)$ sort algorithms
  – Bubble sort
  – Selection sort
  – Insertion sort
  – many, many others...

• **NEVER** use these for large datasets
  – Use $\mathcal{O}(n \log n)$ sorts instead

• But very useful for small datasets
Topic 05: $n^2$ Sorts

- Selection Sort / Loop Invariants
- Insertion Sort
- Bubble Sort
- Time Complexity
Selection Sort

INPUT: arr[]

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

In Class:
Let's simulate this
Loop Invariants

INPUT: arr[]

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Loop Invariants:
What can we say about the array at this point in the code?

What is true in every pass of the loop?
INPUT: arr[]

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Loop Invariant:
The elements arr[0..i] (2nd endpoint exclusive) are the minimums, and are sorted.
Loop Invariants

INPUT: arr[

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Proving a Loop Invariant:
Use induction!

Base: Before the first pass
Inductive: Keeps going
Loop Invariants

INPUT: arr[

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Conjecture:
The elements arr[0..i] (2\textsuperscript{nd} endpoint exclusive) are the minimums, and are sorted.

Base case:
Vacuously true, since the slice arr[0..0] has zero elements.
Loop Invariants

**Inductive case:**
Assume that the invariant holds at the beginning of a pass of the loop; prove it still holds at the end.

In the loop, we scan the range of *unsorted* data, looking for the minimum. We find the minimum of this range, and add it just after the previously-sorted range.

By the I.H., the data *before* this element was already the minimum, and sorted; this element is $\geq$ the topmost element of that range. Moreover, it is clear that it is $\leq$ the minimum element of that which is leftover to the right.

Thus, the loop invariant continues to hold.
Loop Invariants

INPUT: arr[]

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Reviewing the Loop Invariant:
Base case – the invariant holds before the first pass of the loop.
Loop Invariants

INPUT: arr[]

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

Reviewing the Loop Invariant:
Inductive case – if the invariant holds at the beginning of one pass of the loop, then it also holds at the end.
Loop Invariants

INPUT: arr[

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]

What's the Point?
We can use the loop invariant to prove something after the loop ends:

When the loop ends, the array is sorted!
Topic 05: $n^2$ Sorts

- Selection Sort / Loop Invariants
- Insertion Sort
- Bubble Sort
- Time Complexity
Insertion Sort

**INPUT:** arr[]

for i=0 to arr.length-1
    consider element arr[i]
    insert arr[i] into arr[0..i]
    shift over elements >= arr[i]

**In Class:**
Let's simulate this
Insertion Sort

INPUT: arr[]

for i=0 to arr.length-1
    consider element arr[i]
    insert arr[i] into arr[0..i]
    shift over elements \( \geq arr[i] \)

Our Intuition:
The “sorted part” gets larger over time.

How to express this?
**Insertion Sort**

**INPUT:** arr[]

for i=0 to arr.length-1
  consider element arr[i]
  insert arr[i] into arr[0..i]
  shift over elements >= arr[i]

**Loop Invariant:**
The range arr[0..i] is sorted.

Let's prove this in class (using induction).
Topic 05: $n^2$ Sorts

- Selection Sort / Loop Invariants
- Insertion Sort
- Bubble Sort
- Time Complexity
INPUT: arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i,i+1)

In Class:
Let's simulate this
Bubble Sort

INPUT: arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i, i+1)

Our Intuition:
The maximum key “bubbles” to the top of the array; in each pass, the next-highest goes up.
Bubble Sort

INPUT: arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i, i+1)

Loop Invariant:
The range arr[length-i..] is sorted.

Let's prove this in class (using induction).
Bubble Sort

INPUT: arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i, i+1)

For every pass of this loop:
The next-highest value is moved to its proper position.
Bubble Sort

INPUT: arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i, i+1)

In the inner loop:
Find the next-highest value, and move it to its position.
Bubble Sort

**INPUT:** arr[]

repeat arr.length-1 times
  for i=0 to arr.length-2
    if (arr[i] > arr[i+1])
      swap(arr, i,i+1)

**Invariant:**
There are no values to the left of arr[j] which are **more** than arr[j].
Topic 05: $n^2$ Sorts

- Selection Sort / Loop Invariants
- Insertion Sort
- Bubble Sort
- Time Complexity
Time Complexity

- What is the time complexity of Selection Sort?

for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]
What is the time complexity of Selection Sort?

for i=0 to arr.length-1
  find the minimum in arr[i..]
  swap the minimum to arr[i]

This is a $\Theta(n)$ operation.
Is It Really $\Theta(n)$?

for $i=0$ to $\text{arr}.\text{length}-1$
   find the minimum in $\text{arr}[i..]$  
   swap the minimum to $\text{arr}[i]$

Total Cost: \[ n + (n-1) + (n-2) + \ldots + 1 = \frac{n(n+1)}{2} = \Theta(n^2) \]

So yes, this is a $\Theta(n)$ operation (per pass).
Time Complexity

- What is the time complexity of Selection Sort?

```
for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]
```

This is a $\Theta(1)$ operation.
Time Complexity

• What is the time complexity of Selection Sort?

```java
for i=0 to arr.length-1
    find the minimum in arr[i..]
    swap the minimum to arr[i]
```

Thus the cost is $\Theta(n)$ per pass, and $\Theta(n^2)$ in total.
Time Complexity

• What is the time complexity of Insertion Sort?

for i=0 to arr.length-1
  consider element arr[i]
  insert arr[i] into arr[0..i]
  shift over elements >= arr[i]

Θ(n) per pass
Θ(n^2) in total
Time Complexity

- What is the time complexity of Bubble Sort?

repeat arr.length - 1 times
  for j = 0 to arr.length - 1
    if (arr[i] > arr[j])
      swap(arr, i, j)

$\Theta(n)$ per pass

$\Theta(n^2)$ in total
Topic 05: \( n^2 \) Sorts

- Selection Sort / Loop Invariants
- Insertion Sort
- Bubble Sort
- Time Complexity

Summary